Update: 2009-04-04 10:53:55 (Sat)
Edit

MCARaTS User's Guide (Version 0.9.5)

This is a guide for users of MCARaTS (Monte Carlo Atmospheric Radiative Transfer Simulator).

(C) Copyright 2009 MCARaTS project. All rights reserved.

Edit

Contents

1. Introduction
2. Installation
3. Model descriptions
4a. Input to mcarats executable
4b. Input to mcarats executable: Namelist integ -
5. Radiative transfer simulation
6. Output from mcarats executable
7. Post-processing
A single page that includes all documents is available here.

Edit

- 1. Introduction
  - What is MCARaTS
  - 3D radiative transfer
  - MCARaTS softwares

Edit

1. Introduction

What is MCARaTS

MCARaTS is an open-source software package to simulate the three-dimensional radiative transfer (3DRT) in atmosphere-ocean-land system using the Monte Carlo methods. The codes can be applied to simulations of solar energy budget and quasi-observation with optical instruments.
A major purpose to use the software is to simulate accurate radiative quantities as virtual observation data obtained with optical instrument (or human eyes) for cloud-containing atmosphere over land/ocean surface. The other objective is to investigate the three-dimensional radiative transfer within inhomogeneous media.
MCARaTS is a general-purpose radiative transfer model (RTM), which means that input should be optical properties of the atmosphere and surfaces (i.e. extinction coefficients, single scattering albedos, phase functions, and sruface BRDFs). Polarization is not taken into account.

Features of the radiative transfer code can be summarized as follows:
What's good : Easy to use, fast, and parallelized.
Basic algorithm : Forward-propagating Monte Carlo photon transport algorithm
Radiative transfer solvers

- Fully-3-D radiative transfer (F3D)
- Partially-3-D radiative transfer (P3D)
- Independent column approximation (ICA)

Input

- Three-dimensionally inhomogeneous atmosphere
- Inhomogeneous surface modeled by BRDF models
  1. Lambertian model
  2. Diffuse-specular-mixture (DSM) model
  3. Rahman-Pinty-Verstraete (RPV) model
  4. Li-Sparse-Ross-Thick (LSRT) model

Output

- Upward/downward/direct fluxes at layer interfaces (area-averaged fluxes)
- Volume-averaged heating rates
- Radiances possibly with pathlength statistics
  1. Area-averaged radiances at arbitrary layer interfaces
  2. Camera images (angle-average radiances looking from arbitrary points)
- Local fluxes at arbitrary points (pathlength-resolved)

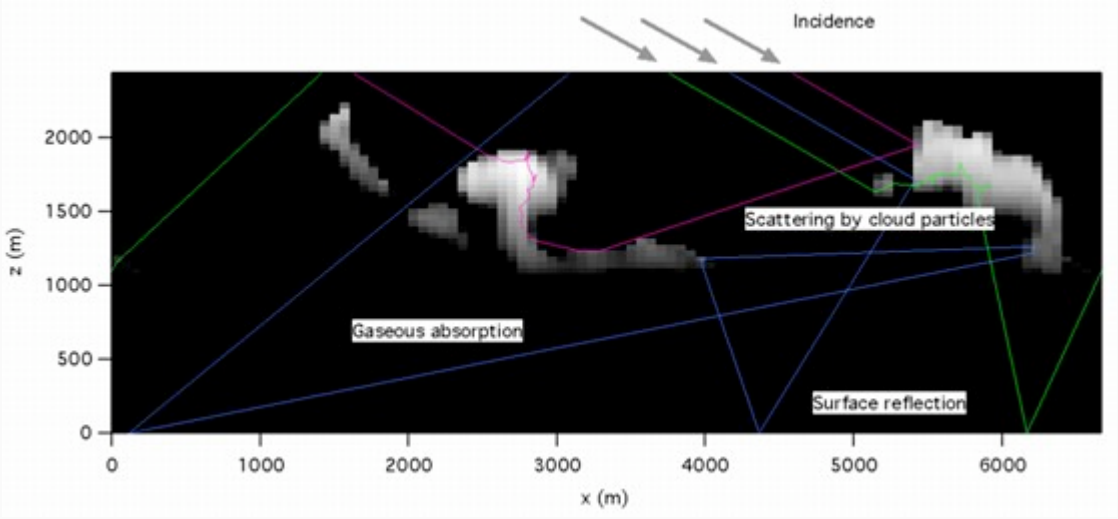
Numerical techniques

- The maximum cross section technique applied to super-cells
- The local estimation method
- Variance reduction techniques
  1. Collision-forcing method for optically-thin media
  2. Flexible truncation of forward peak of phase function
  3. Numerical diffusion
  4. The Russian roulette method
- Parallelization using MPI

Requirements

- Linux/UNIX-like operating system
- Fortran 77 compiler compatible with GCC (GNU's compiler collection)
- MPI is optionally needed for parallelization.

References : Several methods and algorithms used is described in a scientific paper:
Iwabuchi, H., 2006: Efficient Monte Carlo methods for radiative transfer modeling. Journal of the Atmospheric Sciences, 63, 2324-2339.



3D radiative transfer

In previous studies on transfer of energy, several kinds of 3D radiative transfer (3DRT) solvers have been developed, such as finite difference and spherical harmonics methods. We can see a variety of 3DRT codes in the I3RC project for example. The Monte Carlo method is a kind of simple and powerful solvers. Its accuracy is well known due to its physically correct basis and a number of tests. That is why the Monte Carlo method has been frequently used as reference to validate other methods.
For a study on 3DRT, an exact method and two approximations to solve 3D radiative transfer are incorporated in the MCARaTS codes:

- Fully-3-D radiative transfer (F3D)
- Partially-3-D radiative transfer (P3D)
- Independent column approximation (ICA)

ICA neglects horizontal transfer of radiation and applies 1D radiative transfer (1DRT) to atmospheric columns independently. Using the ICA mode, we can compare 3D and 1D schemes strikingly even for very complicated model atmospheres, not being affected by different property descriptions (e.g., constant or trilinear extinction in each cell), area/volume-averaging or method of phase function truncation etc.

MCARaTS softwares

The package includes softwares for the followings:

- Radiative transfer solvers,
- Post-processing the output data.

The post-processing tools process the data file output from the RT solvers. The RT output file is in a unique format, as described later. The processing tools can convert the output data and calculate the basic statistics. See also the corresponding manual pages for details.

Edit

Next

- 2. Installation
  - Compatibility
  - Unpacking
  - Configuration
  - How to install
  - Possible problems
  - Tests
  - Memory tuning
  - Porting to Mac

Edit

2. Installation

Compatibility

MCARaTS v0.9.5 works in UNIX or Linux-like environment. Tested sets are as follows:

- Gfortran, Mac OS X with Intel Core 2 Duo
- G95 (Fortran compiler), Cygwin, Windows
- Intel Fortran Compiler version 9 , Redhat Linux WS3 with Intel Xeon
- Intel Fortran compiler version 9.1 or higher, SuSE Linux, SGI Altix4700

Older versions also worked under the following conditions, but not tested for MCARaTS v0.9.5:

- GCC (GNU Compiler Collection) version 2.95 or higher
- Fujitsu Fortran compiler version 3.0
- UNIX on the HP (Compaq) AlphaServerSC with HP (Compaq) Fortran compiler
- Mac OS X on an Apple PPC Macintosh with GCC version 3.2 or higher
- SUN Solaris on a UltraSPARC workstation with SUN Fortran 77 version 4.2.1.
- UNIX on SGI Origin2000 with SGI Fortran compiler.

Your reports about porting to the other type of computer system are welcome. Reports of computational speed are also welcome.

Unpacking

The softwares are archived to a tar-gzipped file. To unpack it,

```
% gunzip mcarats-*.tar.gz
% tar xf mcarats-*.tar
```

then you will find a directory, mcarats-*, that includes the source codes and sample files as follows:

```
ReadMe   : Read this first
src/     : Source codes
examples/ : Example I/O files
shl/     : Shell scripts.
```

Configuration

In some cases, a few system-dependent configurations must be done before installation.
If the Fortran compiler command is not "f77" and/or you would like to use compiler-specific options, then you have to edit the file:

```
./src/Makefile
./src/*/Makefile
```

If you are building MPI-parallelized codes, you might need to edit a file, ./src/mcarats/Makefile, for system-specific configuration of MPI libraries.

How to install

To build the executable files, execute the following commands:

```
% cd src
% make install
% cd ..
```

After installation you will find executable files in ./bin directory. If you have some problem, see the next section.
To build the parallelization codes, for example,

```
% cd src
% make clean install USE_MPI=1 BINDIR=../bin-mpi
% cd ..
```

Radiative transfer codes are parallelized with MPI, but other codes are not. See corresponding Makefile, src/mcarats/Makefile, for details.

Possible problems

Possible problems in the installation and resolutions are as follows:
1. Non-standard manners for Fortran77
The codes in the package use some extensions of Fortran 77, but they are common, and some of them are compatible with Fortran90.

- Namelist input
- "do/end do" loop,
- "getarg()" to get arguments,
- "itime()" to get the time. It is used to initialize random number,
- Long variable name.

If you had some problems, a possible reason is the compatibility related to the above non-standard functions. Some compilers may require specific options to make the above extensions to work. For example, Intel Fortran compiler v7 requires "-Vaxlib" option to compile "getarg" and "itime" extensions (higher versions of Intel compiler does not require such a special option).
2. Bug?
Any bug report will be appreciated.

Tests

It is highly recommended for users to test the codes for the sample cases. For that purpose, a shell script is provided.

```
% ./examples/Job.csh
```

The results will be found in ./case[1-3] directories. They can be compared with sample files in ./samples/case[1-3]. Using larger number of photons for simulation, you may get more accurate result that might be closer to the sample files. The number of photons are defined in the shell script. Note that the result will differ second by second, because the random number for Monte Carlo simulation is initialized with a time the user starts the simulation. Usages and functions of the codes are easily learned from the tests.

Memory tuning

The memory size used for the executables can be tuned in the followings, before the installation:

```
./src/mcarats/mcarats1d.inc
./src/mcarats/mcarats2d.inc
./src/mcarats/mcarats3d.inc
./src/process/process.inc
```

By default, the codes require about 350 MB at maximum for memory.

Porting to Mac

Mac OS X provides a good platform for numerical computing with gcc and some "Xcode" tools (make and ar etc.) that are included in "Developer tools". But any Fortran compiler is not included by default. To compile Fortran source code, there is a good guide webpageproviding binary packages of Fortran compilers (g77) for Mac OS X. One should follow instructions written in the above webpage (installation of Developer tools and the g77 package, operating ranlib command). Using them, one of the authors tried the computation on Mac OS X 10.2 and 10.3. The gcc (g77) versions 3.3 and 3.4 were usable. Recommended compile options are "-O -ffast-math -I.". GCC g77 binaries are also available from Fink.

Edit

Next

- 2. Installation
  - Compatibility
  - Unpacking
  - Configuration
  - How to install
  - Possible problems
  - Tests
  - Memory tuning
  - Porting to Mac

Edit

2. Installation

Compatibility

MCARaTS v0.9.5 works in UNIX or Linux-like environment. Tested sets are as follows:

- Gfortran, Mac OS X with Intel Core 2 Duo
- G95 (Fortran compiler), Cygwin, Windows
- Intel Fortran Compiler version 9 , Redhat Linux WS3 with Intel Xeon
- Intel Fortran compiler version 9.1 or higher, SuSE Linux, SGI Altix4700

Older versions also worked under the following conditions, but not tested for MCARaTS v0.9.5:

- GCC (GNU Compiler Collection) version 2.95 or higher
- Fujitsu Fortran compiler version 3.0
- UNIX on the HP (Compaq) AlphaServerSC with HP (Compaq) Fortran compiler
- Mac OS X on an Apple PPC Macintosh with GCC version 3.2 or higher
- SUN Solaris on a UltraSPARC workstation with SUN Fortran 77 version 4.2.1.
- UNIX on SGI Origin2000 with SGI Fortran compiler.

Your reports about porting to the other type of computer system are welcome. Reports of computational speed are also welcome.

Unpacking

The softwares are archived to a tar-gzipped file. To unpack it,

```
% gunzip mcarats-*.tar.gz
% tar xf mcarats-*.tar
```

then you will find a directory, mcarats-*, that includes the source codes and sample files as follows:

```
ReadMe   : Read this first
src/     : Source codes
examples/ : Example I/O files
shl/     : Shell scripts.
```

Configuration

In some cases, a few system-dependent configurations must be done before installation.
If the Fortran compiler command is not "f77" and/or you would like to use compiler-specific options, then you have to edit the file:

```
./src/Makefile
./src/*/Makefile
```

If you are building MPI-parallelized codes, you might need to edit a file, ./src/mcarats/Makefile, for system-specific configuration of MPI libraries.

How to install

To build the executable files, execute the following commands:

```
% cd src
% make install
% cd ..
```

After installation you will find executable files in ./bin directory. If you have some problem, see the next section.
To build the parallelization codes, for example,

```
% cd src
% make clean install USE_MPI=1 BINDIR=../bin-mpi
% cd ..
```

Radiative transfer codes are parallelized with MPI, but other codes are not. See corresponding Makefile, src/mcarats/Makefile, for details.

Possible problems

Possible problems in the installation and resolutions are as follows:
1. Non-standard manners for Fortran77
The codes in the package use some extensions of Fortran 77, but they are common, and some of them are compatible with Fortran90.

- Namelist input
- "do/end do" loop,
- "getarg()" to get arguments,
- "itime()" to get the time. It is used to initialize random number,
- Long variable name.

If you had some problems, a possible reason is the compatibility related to the above non-standard functions. Some compilers may require specific options to make the above extensions to work. For example, Intel Fortran compiler v7 requires "-Vaxlib" option to compile "getarg" and "itime" extensions (higher versions of Intel compiler does not require such a special option).
2. Bug?
Any bug report will be appreciated.

Tests

It is highly recommended for users to test the codes for the sample cases. For that purpose, a shell script is provided.

```
% ./examples/Job.csh
```

The results will be found in ./case[1-3] directories. They can be compared with sample files in ./samples/case[1-3]. Using larger number of photons for simulation, you may get more accurate result that might be closer to the sample files. The number of photons are defined in the shell script. Note that the result will differ second by second, because the random number for Monte Carlo simulation is initialized with a time the user starts the simulation. Usages and functions of the codes are easily learned from the tests.

Memory tuning

The memory size used for the executables can be tuned in the followings, before the installation:

```
./src/mcarats/mcarats1d.inc
./src/mcarats/mcarats2d.inc
./src/mcarats/mcarats3d.inc
./src/process/process.inc
```

By default, the codes require about 350 MB at maximum for memory.

Porting to Mac

Mac OS X provides a good platform for numerical computing with gcc and some "Xcode" tools (make and ar etc.) that are included in "Developer tools". But any Fortran compiler is not included by default. To compile Fortran source code, there is a good guide webpageproviding binary packages of Fortran compilers (g77) for Mac OS X. One should follow instructions written in the above webpage (installation of Developer tools and the g77 package, operating ranlib command). Using them, one of the authors tried the computation on Mac OS X 10.2 and 10.3. The gcc (g77) versions 3.3 and 3.4 were usable. Recommended compile options are "-O -ffast-math -I.". GCC g77 binaries are also available from Fink.

Edit

Next

- 3. Model descriptions
  - Algorithms
  - Geometry
  - Surface
  - Atmosphere
  - Integrators
  - Methods and techniques

Edit

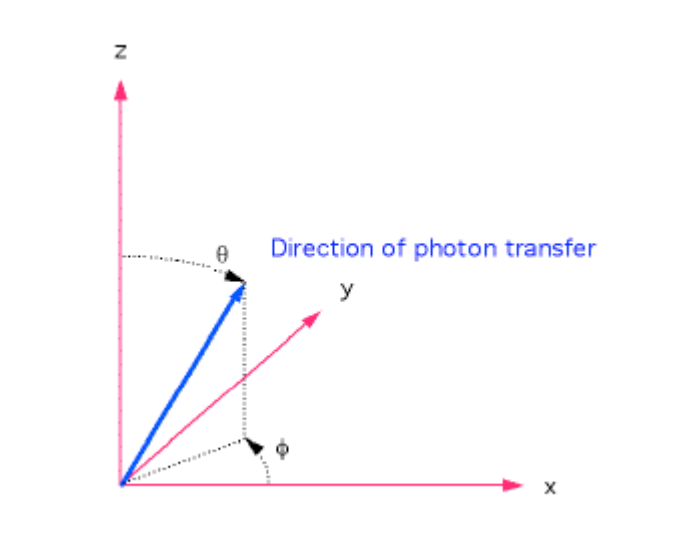
3. Model descriptions

Algorithms

Used algorithms are described in Iwabuchi (2006, J. Atmos. Sci.). The model uses Monte Carlo methods for simulating photon trajectories and samples fluxes, heating rates, radiances, and other radiometric quantities. The local estimation method is used for radiance averaged over some specific area or over specific solid angle. The maximum cross section method is used for acceleration of photon tracing in inhomogeneous media. Other methods useful for variance reduction are also used and documented in the scientific paper. The flowing figure is an algorithm flow chart.

Geometry

The position of photon is defined in the 3D Cartesian coordinate system. The direction of photon transport is defined by zenith angle (theta) and azimuth angle (phi) from the X-axis, as in the following figure.



The incident and emergent directions for pixel radiances can be specified by the user. The source zenith and azimuth angles are defined as theta0 and phi0, respectively. Similarly, emergent zenith and azimuth angles are defined as theta1 and phi1, respectively. The x-, y- and z-coordinate is defined in 0– xmax, 0– ymaxand zmin– zmax, respectively. The horizontal boundary condition is cyclic. The vertical boundaries z=zmin and zmax correspond to bottom and top, respectively, of the atmosphere (BOA and TOA, respectively).

Surface

The surface is modeled as one of four models

- black (no reflection)
- Lambertian model
- Diffuse-specular mixture model: diffuse reflection is Lambertian, and specular reflection is Fresnellian for rough surface facets.
- Rahman-Pinty-Verstraete BRDF model
- Li-Sparse-Ross-Thick BRDF model

Surface model and its parameters can vary two-dimensionally.

Atmosphere

The model atmosphere is divided into cell volumes three-dimensionally. The number of layer is defined as nz.

The user can specify cloud layers ("3D layers") arbitrarily with horizontal inhomogeneity. Other layers are assumed to be horizontally homogeneous. The 3D layers are divided into cells in the vertical and horizontal directions. The numbers of 3D cells in x-, y-, and z-directions are defined as nx, ny, and nz3, respectively. Modeled atmospheric media include the followings:

- Scattering media: For each one medium, extinction coefficient, single scattering albedo and a ID number of definition of phase

function are arbitrarily specified. The user defines some kinds of phase function. Properties can be specified in 1D (for layers) and/or in 3D (for 3D cells):

1. three kinds of media specified for each 1D layer
2. several kinds of media specified for each 3D cell

- Gaseous absorbing media: Modeled by correlated k-distribution. The absorption coefficients are specified for both 1D layers and 3D cells

Integrators

Various radiative quantities can be computed by the code:

- Upward/downward/direct fluxes at layer interfaces (area-averaged fluxes)
- Voxel-averaged 3-D heating rates
- Pixel-averaged radiances at arbitrary layer interfaces
- Camera images (angle-average radiances looking from arbitrary points)
- Local fluxes at arbitrary points (pathlength-resolved)
- Average pathlengths normalized by vertical layer depths (so-called air-mass factor)

The area-averaged flux detectors are specified at layer boundaries for upward, downward and direct fluxes. The flux is computed as pixel-averaged quantity.
The area-averaged radiances at layer boundaries are defined by the user with zenith and azimuth angles (therdcand phirdc, respectively). The numbers, nxrand nyr, of pixels along x- and y-axes are given by the user. That definition is independent of flux.

Methods and techniques

Various methods and techniques are used in the code, for better performance. Some of them are described in Iwabuchi (2006, J. Atmos. Sci.).

- The maximum cross section technique applied to super-cells
- The local estimation method
- Variance reduction techniques
- Collision-forcing method for optically-thin media
- Truncation approximations for forward-peaked phase functions
- Numerical diffusion
- The Russian roulette method
- Parallelization using MPI

The method of truncation of forward peak of phase function is incorporated. The truncation fraction adaptively increases with the order of scattering. The user specifies the number of truncation regimes (nchi), light-diffusivity thresholds (chihi & chilo), and maximum truncation factor (fmax).

- 4. Input to "mcarats" code
  - Full list of input parameters
  - Notes on namelist input
  - Namelist src
    - jnorm
    - nsrc
    - jdefsrc
    - wlen
    - dwlen
    - fsrc
    - dqsrc
    - thesrc, phisrc
    - jphisrc
    - xsrc, ysrc, zsrc
    - diasrc
    - Note on output quantities and physical units
  - Namelist tech
    - nbyte
    - nsss
    - nsmax
    - nstrun
    - jtech
    - chihi
    - chilo
    - fmax
    - nchi
    - wmin, wmax
    - wfac
    - nrr
    - taumin
    - dtaumin
    - rlcf
    - nlcf
    - zetamin
    - diff0, diff1
    - difh0, difh1
    - difr0, difr1
    - difc0, difc1
    - nudsm
    - nurpv, nulsrt
    - nqsfc
    - srmin
    - srexp
    - emtmin
    - fsupg
    - fsupi
    - nzzmax
    - Note: Approximation or not?
    - Note on transformations of scattering phase functions

Edit

4. Input to "mcarats" code

Executables for radiative transfer simulation are named as "mcarats?d" (?=1, 2, or 3). We call them "mcarats codes" hereafter. All input data of mcarats codes can be given as Fortran namelist variables.

The built-in namelists are as follows:
src : radiation source definitions
tech : technical parameters including tuning parameters for efficiency
integ : integrator (fluxes, heating rates, radiances, camera images, etc) configurations
model : general model configurations
mdlphs : phase function model configurations
mdlsfc : surface model configurations
mdla3d : 3-D atmospheric model configurations

Some large array data such as scattering phase functions and surface and 3-D atmospheric properties can be given by external files:
filephs : phase function model data
filesfc : surface model data
filea3d : 3-D atmosphere model data

The filenames are given in the namelists, mdlphs, mdlsfc, and mdla3d, respectively. The following subsections describes details of each input parameters for the namelists and the external files. There are many input parameters. In addition, this document is not complete. To learn how to set up the parameters for the user's purpose, the easiest way might be to explore the example cases, for which input data files are included in the package. Three cases are provided:

- Case 1: three plane-parallel layers, flux and radiance calculations
- Case 2: 3-D inhomogeneous case, broken cumulus-type clouds, inhomogeneous diffuse-specular-mixture BRDF model, fluxes and heating rates
- Case 3: 3-D inhomogeneous case, overcast stratocumulus cloud, homogeneous Lambertian or Rahman-Pinty-Verstraete BRDF model
  - Remote sensing ("rs") experiment: calculation of pixel-averaged, reflected radiances at the top of atmosphere (TOA)
  - Camera image ("ci") experiment: calculation of angle-averaged radiances looking up the sky from a point on the surface
  - Heating rate ("hr") experiment: calculation of 3-D distributions of pixel-averaged fluxes and volume-averaged heating rates
  - Point source ("ps") experiment: calculation of narrow FOV fluxes looking up the sky from a point on the surface; the radiation source is artificial laser beam emitted vertically near the detector.

Full list of input parameters

Namelist src

```
wlen, dwlen, fsrc, dqsrc, thesrc, phisrc, xsrc, ysrc, zsrc, diasrc, jdefsrc, jphisrc,
nsrc, jnorm
```

Namelist tech

```
nbyte, nsss, nstrun, nsmax, jtech, wmin, wmax, wfac, nrr, chihi, chilo, fmax, nchi,
diff0, diffl, difh0, difhl, difr0, difrl, difc0, difcl, taumin, dtaumin, emtmin, zetamin,
rlcf, nlcf, fsupg, fsupi, nzzmax, nqsfc, nudsm, nurpv, nulsrt, srmin, srexp,
(pnmin, omgopq, chiiso, trsmin, nxxs, nyys, nzzs, nsle)
```

Namelist integ

```
jsflx, jshrt, nrdc, therdc, phirdc, jzrdc, nxr, nyr, ncam, nxc, nyc, xcam, ycam, rmincam,
rmidcam, rmaxcam, jzcam, thecam, phicam, ahcam, avcam, amcam, diacam, npot, nppot, nqpot,
jsppot, xpot, ypot, rminpot, rmidpot, rmaxpot, jzpot, thepot, phipot, pmaxpot, pminpot,
qmaxpot, diapot
```

Namelist model

```
nx, ny, nz, iz3l, nz3, xbin, ybin, zgrd, jrfr, jsph, rplanet, jtmplev, tmpald, extlst,
omglst, jpflst, ext2nd, omg2nd, jpf2nd, ext3rd, omg3rd, jpf3rd, nkd, wkd, absgld, dens,
drfr
```

Namelist mdlphs

```
filephs, npf, nang, ang, phs
```

Namelist mdlsfc

```
filesfc, tmps2d, jsfc2d, psfc2d, nxb, nyb
```

Namelist mdla3d

```
filea3d, ifmt3d, npar, tmpa3d, absg3d, extp3d, omgp3d, jpfp3d
```

Notes on namelist input

The namelist function is a feature of Fortran90, while most Fortran77 compilers (including the GNU compilers) have extensions to read/write namelists. The namelists for mcarats input include many parameters, but the the user does not need to specify all variables' values. This feature makes the Fortran's namelist flexible and easy to use. For example, when the user gives no value for every variables, then default initialization data will be used. It is the case 1 of the examples, where the content of the input file is as the followings:

```
&src /
&tech /
&integ /
&model /
&mdlphs /
&mdlsfc /
&mdla3d /
```

This is the case as the sample case 1 (see examples/case1/conf), in which flux profile is simulated for a plane parallel weakly-absorbing cloud. Default initialization data are defined in ./src/mcarats/artdinput.inc, and they are customizable before compiling.

The user can partially set values for one variable in the namelist. For example, a sentence,

```
absg1d(1:10, 1)=4.0e-5,  3.0e-5,  2.0e-5,  1.0e-5,  5*1.0e-6
```

sets values for a part of the 2-D array variable absg1d(iz,ikd), which may be declared as a very large array in the source code.

Another feature of Fortran namelist is a freedom of ordering of each member variable in a namelist block. Any order is acceptable in one namelist block. For details, see Fortran language reference for the compiler.

Namelist src

jnorm

[integer scalar]
When jnorm=1, all output radiative quantities are normalized by domain-average source flux, as follows:

- area-averaged fluxes: F/F0
- volume-averaged heating rates: H/F0
- area-averaged radiances: pi*I/F0
- angle-averaged local radiances (camera image): pi*I/F0
- local plane/spherical fluxes: F/F0

When the source is collimated solar beam (jdefsrc=1) alone, for example, the domain-averaged source flux is

```
F0 = fsrc * abs(cos(180 - thesrc))
```

where fsrc is the incident irradiance on the plane normal to the solar beam, and thesrc is the zenith angle of the incoming beam.

nsrc

[integer scalar]
Number of source definitions.

jdefsrc

[integer 1-D array, {dat(isrc), isrc=1, nsrc}]
Defines external and/or internal sources, as follows:

- 0: localized sources (the location, direction, and angular width are specified by the user)
- 1: solar sources (incidence from the top of atmosphere)
- 2: solar plus thermal emission sources
- 3: thermal emission sources.

wlen

[real 1-D array, {dat(isrc), isrc=1, nsrc}]
Wavelength (micrometer) of light for thermal emission. This is used only when jdefsrc = 2 or 3. If the band width (dwlen) is not zero, then wlen is the lower limit of the band. See also dwlen.

dwlen

[real 1-D array, {dat(isrc), isrc=1, nsrc)}]
Wavelength band width (micrometer). Null is acceptable. When dwlen=0, the simulation is completely monochromatic, and output quantities are spectral ones. Otherwise (dwlen > 0), output quantities are also spectrally integrated ones, and spectrally integrated thermal emission are calculated (jdefsrc=2 or 3).

fsrc

[real 1-D array, {dat(isrc), isrc=1,nsrc}]
Spectral source irradiance or power.

- When the solar source is present (jdefsrc = 1 or 2), fsrc is the solar source spectral irradiance (W/m^2/micrometer) at the top of atmosphere. The fsrc is defined as integrated on the plane normal to the center direction of the source emission. Note that the definition differs from the spherical flux (or mean radiance). The top-of-atmosphere solar source irradiance on the horizontal plane should be, for collimated solar incidence (dqsrc=0),

```
fsrc*abs(cos(180 - thesrc))
```

- When the source is localized (jdefsrc = 0), fsrc is the emitted spectral power (W/micrometer).

Even if dwlen > 0, fsrc should be spectral quantity, averaged over the band. Then spectrally integrated solar source irradiance should be dwlen*fsrc, for example.
If jnorm=1, then calculated results will be normalized, and thus any value for fsrc should produce the same output.

dqsrc

[real 1-D array, {dat(isrc), isrc=1,nsrc}]
Half cone angle (in degree) that determines angular width of solar or localized source, used only when jdefsrc = 0, 1 or 2. This should be 0.2665666 (degree) for solar incidence at an annual-mean state of the sun-earth system. Give zero for cases in which sources are negligibly small or infinitely far.

thesrc, phisrc

[real 1-D array, {dat(isrc), isrc=1,nsrc}]
Zenith and azimuth angles (in degree), respectively, of the source emission direction vector. Used only when jdefsrc = 0, 1 or 2. Note the source direction is defined as transport direction of source light. If the source is solar light, then the transport direction corresponds to the sun-to-surface vector. Thus, the solar zenith angle should be (180 - thesrc).

jphisrc

[integer 1-D array, {dat(isrc), isrc=1, nsrc}]
If jphisrc = 0, then the azimuth angles of the source vectors are set as phisrc. Else if jphisrc = 1, then the azimuth angles are set as random between 0 to 360 degrees.

xsrc, ysrc, zsrc

[real 1-D array, {dat(isrc), isrc=1,nsrc}]
Coordinates that specifies the location of the localized source, used when jdefsrc=0.

diasrc

[real 1-D array, {dat(isrc), isrc=1,nsrc}]
Diameter of the emitting source surface (or the surface of the outmost lens of the artificial source), when the source is localized (jdefsrc = 0). The localized source should be very small compared to the model domain, but non-zero finite size is allowed. If diasrc = 0, then it means the source is from a point. Otherwise, the source surface is assumed to be spherical. This parameter, diasrc, is related to dqsrc and the diameter (D) of the emitting sphere, as follows:

```
diasrc = D * sin(dqsrc)    if dqsrc < 90 degrees
diasrc = D                 otherwise
```

Note on output quantities and physical units

- jnorm=1: Output quantities are all normalized as described above.
- jnorm=0: Output quantities have physical units.

a) dwlen=0: spectral, monochromatic calculations, as the followings

- area-averaged spectral fluxes (e.g., W/m^2/micrometer),
- volume-averaged spectral heating rates (e.g., convergences, W/m^3/micrometer),
- area-averaged spectral radiances (e.g., W/m^2/steradian/micrometer),
- angle-averaged, local spectral radiances (e.g., W/m^2/steradian/micrometer), and
- local plane/spherical fluxes (e.g., W/m^2/micrometer)

b) dwlen > 0: spectrally-integrated calculations; Units do not include "1/micrometer".

Namelist tech

nbyte

[integer scalar]
# of length of single binary record for one "real*4" value in direct access format files. Used when an external input file for 3-D model data is recorded in direct-access format (ifmt3d=1). In other cases (ifmt3d=0), this variable is not used at all. An appropriate value for nbyte is computer's architecture and compiler dependent (usually 4 or 1). If you use the GCC g77 compiler on a x86 PC, nbyte should be 4. If you use Intel Fortran compiler (on x86 PC or other machines), then nbyte should be 1. See also ifmt3d.

nsss

[integer scalar]
Order of collision that determines RT scheme shift when the partially-three-dimensional (P3D) RT solver is selected. The solver selection is done when the user run the mcarats executables (see here for details). After a scattering or reflection event of the order of nsss, the photon packet never moves in horizontal direction, i.e., independent column approximation (ICA). In other solvers, this parameter has no effect on simulations.

nsmax

[integer scalar]
Maximum order of collision. After the scattering of the order, energy of photon packet will be forced to null, and the simulation for that packet will be terminated. This might be useful for optically thick, weakly-absorbing atmosphere. For precision, however, this parameter should be set at a very large value (about 10000 or larger).

nstrun

[integer scalar]
Maximum order of collision for non-approximated radiative transfer simulation. For collision events with order less than or equal to nstrun, no truncation approximation is used. The approximation can introduce (usually very small) bias. Thus this parameter might be useful for some purposes. For example, if the user need to get accurate separation of direct-beam and multiple scattering, then setting as nstrun = 1 forces the direct beams non-approximated and then for multiple scattering the truncation approximations can be used.

jtech

[integer scalar]
Switch to select one of built-in presets for technical parameters described in the followings. Many parameters in the /tech/ namelist might be difficult for non-power users to tune. Presets were well tuned by the developer. The preset can be selected by specifying jtech, as follows:

```
-1: No preset is used.  the user may specify all /tech/ variables' values.
0: No technique that can improve numerical efficiency is used.  This might be useful for debugging and/or tuning experiment
1: Preset for high-quality radiances of out-of-aureole directions
2: Preset for high-quality radiances of inner-aureole directions
3: Preset for low-quality radiances
4: Preset for high-quality flux/heating-rates
5: Preset for low-quality flux/heating-rates
6: Preset for good-looking camera images using quick-and-dirty methods
```

For "low-quality" calculations, quick-and-dirty methods are used. The quality setting can differ by whether the user's goal is accuracy or computation speed.
If jtech is not -1, then all values given in the input file are overwritten by the built-in preset. The following parameters are set automatically:

```
chilo, chihi, fmax, nchi, wmin, wmax, wfac, nrr, taumin, dtaumin, rlcf, nlcf, zetamin,
diff0, diff1, difh0, difh1, difr0, difr1, difc0, difc1, nudsm, nurpv, nulsrt, nqsfc,
srmin, srexp, emtmin, fsupg, fsupi, nzzmax
```

The preset values are given in src/mcarats/artsettech.f.

The followings in the namelist tech can be automatically set by jtech

chihi

[real scalar]
Upper limit of the directionality parameter, chi, used for the dual-end truncation approximation (Iwabuchi, 2006, JAS). The parameter chi is 1 for collimated radiation (solar direct beam) and 0 for isotropic light. For chi > chihi, no truncation approximation is used. The truncation fraction linearly increases with decreasing chi from chihi to chilo. Setting as 0.6 < chihi < 0.9 is recommended. See also chilo and fmax.

chilo

[real scalar]
Lower limit of chi for regime shift in truncation approximations. For chi < chilo, the truncation fraction is fixed at each value prescribed for each scattering phase function. Setting as 0.2 < chilo < 0.6 is recommended. See also chihi and fmax.

fmax
[real scalar]
Scaling factor determining maximum delta-fraction of the dual-end truncation approximations, for chi larger than or equal to chilo. The approximation reduces computation time of fluxes and heating rates by 10-80% and also significantly improves accuracy of computed radiances by reducing noises by a factor of 10 (for cloud reflectance at visible wavelengths). Recommended value is between 0.6 and 0.8. See also chihi and chilo.

nchi
[integer scalar]
Number of truncation approximation regimes, one of which corresponds to one range of chi. Should be larger than or equal to 3, i. e., at least three regimes are used:

- chi > chihi: no truncation approximation
- chiiso < chi < chihi: dual-end truncation approximation
- chi < chiiso: delta-isotropic approximation

wmin, wmax
[real scalar]
Minimum and maximum limits, respectively, for ideal weight of single photon packet. These limits avoid large dispersion in the photon weight.

wfac
[real scalar]
Factor to modify the ideal weight of photon packet. The ideal weight is multiplied by wfac once every nrr collision events. If the weight of photon packet is less than the ideal weight, then the Russian roulette method is applied and photon population is reduced. Thus survived photon packets have increased weight that is the same as the ideal weight. See also nrr.

nrr
[integer scalar]
The ideal weight is multiplied by wfac once every nrr collision events.

taumin
[real scalar]
Minimum threshold of vertical optical thickness of the total atmosphere. If actual optical thickness is less than this threshold, then the collision forcing method is used, and the scaled value of the total optical thickness is set to taumin. Recommended value is between 2 and 10.

dtaumin
[real scalar]
Minimum threshold of vertical optical thickness of single layer. If actual optical thickness of one layer is less than this threshold, then the collision-forcing method is used in the layer, and the scaled value of the optical thickness is set to dtaumin. Thus collision events are forced to occur with prescribed probability even in very optically thin layer. Usually recommended value is 0.01 for heating rate calculation.

rlcf
[real scalar]
Scale length (m) for Local Collision-Forcing (LCF), which forces artificial collisions at locations near the cameras. Extinction coefficients are scaled so as to be large near the camera. See also nlcf.

nlcf
[integer scalar]
Power exponent for LCF. Should be 1 or 2 usually.
Scaled extinction coefficients (B_ext1) will be

```
B_extl = max(B_min, B_ext)
```

for original extinction coefficient B_ext, and the min extinction coefficient is

```
B_min = dtaumin / dz_lay * (zdif / rlcf)**nlcf
```

where dz_lay is a layer geometrical thickness and zdif is height difference from the camera location. See also rlcf and dtaumin.

zetamin
[real scalar]
Minimum threshold for each energy contribution sampled by the local estimation method, which is used for calculating area-averaged radiances, angle-averaged local radiances (the camera image), and local fluxes. Radiance contribution from each emission/scattering/reflection event is calculated as the normalized angular distribution function multiplied by transmittance

between the event point and the detector. Recommended value is 0.3.

diff0, diff1
[real scalar]
Artificial diffusion coefficients for flux sampling. Recommended setting is as diff0=1 and diff1=0.1.

difh0, difh1
[real scalar]
Artificial diffusion coefficients for heating rate sampling. Recommended setting is as difh0=1 and difh1=0.1.

difr0, difr1
[real scalar]
Artificial diffusion coefficients for area-averaged radiance sampling. Recommended setting is as difr0=50 and difr1=0.01.

difc0, difc1
[real scalar]
Artificial diffusion coefficients for angle-averaged local radiance (camera) sampling. Recommended setting is as difc0=50 and difc1=0.01.

nudsm
[integer scalar]
Number of solar zenith angles for the look-up tables (LUTs) for diffuse-specular-mixture (DSM) BRDF model. The LUTs contains parameters for albedo calculation and determination of reflection direction. In simulations, the parameters used are interpolated from the LUT. Thus larger nudsm value achieves higher accuracy but requires larger memory for the LUTs.

nurpv, nulsrt
[integer scalar]
Same as nudsm, but for Rahman-Pinty-Verstraete (RPV) and Li-Sparse-Ross-Thick (LSRT) BRDF models, respectively.

nqsfc
[integer scalar]
Number of quadrature points for surface albedo calculations for the BRDF models, DSM and RPV models. Used when creating the LUTs for albedo, which is calculated by integration of the BRDF over the hemisphere. Larger quadrature points can achieve accurate results but may take longer CPU time. Usually 10 is enough to achieve high accuracy as 0.5% (relative unit).

srmin
[real scalar]
Parameter for determination of reflection direction by the BRDF models, DSM and RPV models. Smaller value needs less computer time, but the determined reflection direction is more approximate. If srmin is very large (>1.0e+8), then no such an approximation is used, and the simulation may require nearly infinite CPU time! Should be larger than or equally to 1. Usually 4 is recommended.

srexp
[real scalar]
Parameter for determination of reflection direction by the BRDF models, DSM and RPV models. Smaller value needs less computer time but the determined reflection direction is more approximate. If srexp=1, then no such an approximation is used, and the simulation may require nearly infinite CPU time! Should be between 0 and 1. Usually 0.5 is recommended.

emtmin
[real scalar]
Minimum threshold for photon populations of thermal emission in single layer. Used only when jdefsrc=2 or 3. This forces to emit photons with prescribed probability even from very optically-thin layer. Should be between 0 and 1. Recommended setting is emtmin=0.01. If emtmin=0, then no such an optimization is used.

fsupg
[real scalar]
Geometrical threshold parameter for automatic super-cell generation for maximum cross section method. Usually should be about 2. See also fsupi.

fsupi
[real scalar]
Inhomogeneity threshold parameter for automatic super-cell generation for maximum cross section method. An optimal value would be 1.3. A super-cell is generated by merging cells util one of conditions are met.

nzzmax
[integer scalar]

Possible max mumber of layers merged into a single super-cell, in which the maximum cross section method is used.

Note: Approximation or not?

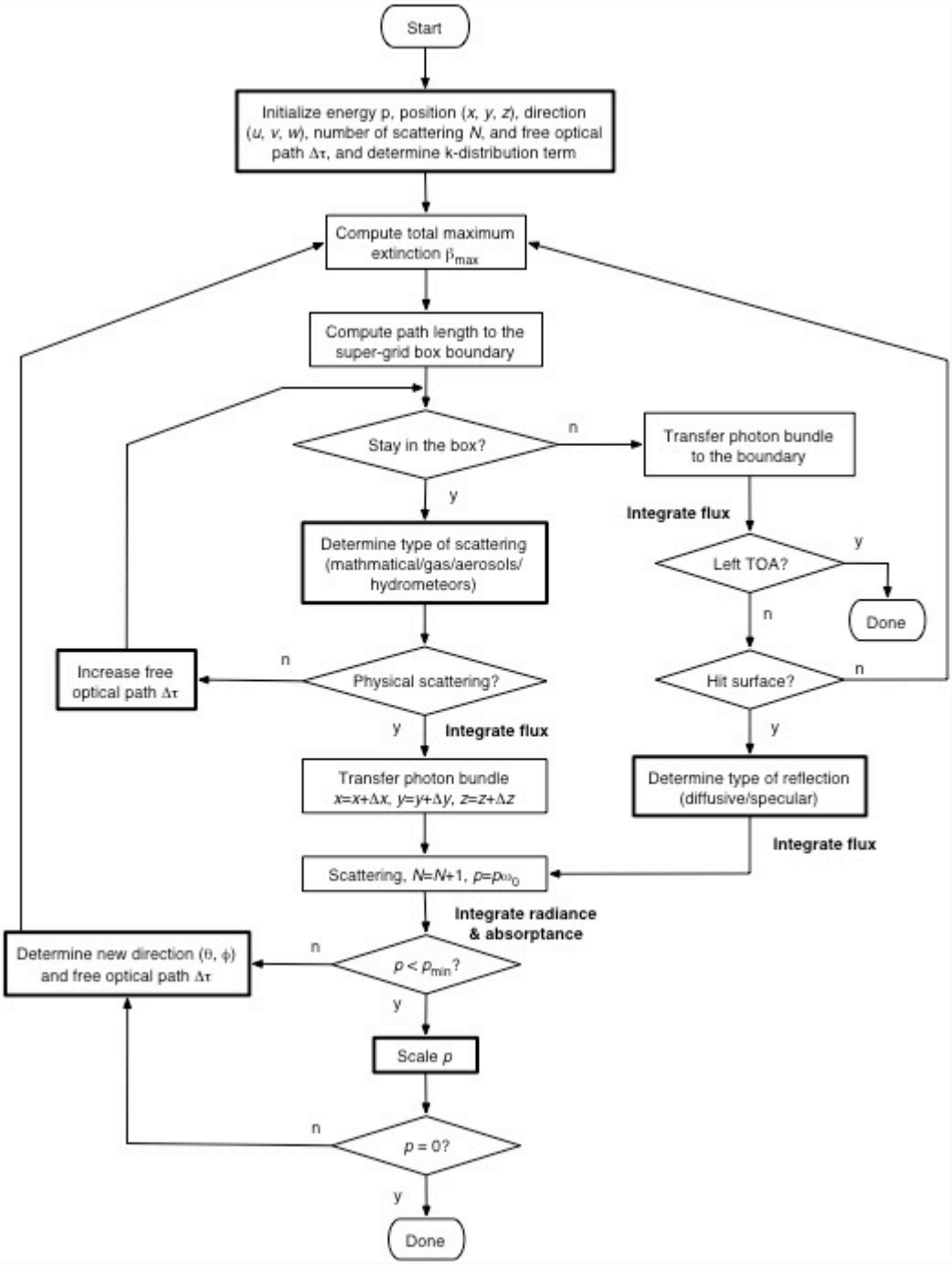
Some parameters in the namelist "tech" activate approximations which may improve numerical efficiency (accuracy vs computation time). However the approximations may introduce bias error. If the user would not like to use any approximation (that can be biasing) for some reason, then a required configuration is as jtech=0, and no biasing method will be used (unbiasing methods are still used).

Note on transformations of scattering phase functions

Transformations of scattering phase functions (so-called truncation approximation) will be used according to values for chihi, chilo, chiiso, and fmax. Note that the transformation will speed computation and reduce Monte Carlo random noises but produce (usually small) bias error, so there is 'trade-off' of accuracy and CPU time. The recommended parameter set for general purpose can be selected by jtech.

Edit

Next


Start
Initialize energy p, position (x, y, z), direction (u, v, w), number of scattering N, and free optical path Δτ, and determine k-distribution term
Compute total maximum extinction β_max
Compute path length to the super-grid box boundary
Stay in the box?
n
Transfer photon bundle to the boundary
y
Integrate flux
Determine type of scattering (mathmatical/gas/aerosols/ hydrometeors)
Left TOA?
y
Done
n
Increase free optical path Δτ
n
Physical scattering?
Hit surface?
n
y
Integrate flux
y
Transfer photon bundle x=x+Δx, y=y+Δy, z=z+Δz
Determine type of reflection (diffusive/specular)
Integrate flux
Scattering, N=N+1, p=pω0
Integrate radiance & absorptance
Determine new direction (θ, ϕ) and free optical path Δτ
n
p < p_min?
y
Scale p
n
p = 0?
y
Done

Edit

Next

- Namelist integ
  - jsflx
  - jshrt
  - nrdc
  - nxr, nyr
  - therdc, phirdc
  - jzrdc
  - ncam
  - nxc, nyc
  - jzcam
  - xcam, ycam
  - rmincam
  - rmidcam
  - rmaxcam
  - thecam, phicam
  - ahcam
  - avcam
  - amcam
  - diacam
  - npot
  - nppot
  - nqpot
  - jsppot
  - jzpot
  - xpot, ypot
  - rminpot
  - rmidpot
  - rmaxpot
  - thepot, phipot
  - pminpot, pmaxpot
  - qmaxpot
  - diapot
- Namelist model
  - nx, ny, nz
  - iz3l
  - nz3
  - xbin, ybin
  - jrfr
  - jsph
  - rplanet
  - jtmplev
  - zgrd
  - drfr
  - dens
  - tmpa1d
  - ext1st, omg1st
  - jpf1st
  - ext2nd, omg2nd, jpf2nd
  - ext3rd, omg3rd, jpf3rd
  - nkd
  - wkd
  - absg1d

- Namelist mdlphs
  - filephs
  - npf
  - nang
  - ang
  - phs
- Namelist mdlsfc
  - filesfc
  - nxb, nyb
  - tmps2d
  - jsfc2d
  - psfc2d
- Namelist mdla3d
  - ifmt3d
  - npar
  - filea3d
  - tmpa3d
  - absg3d
  - extp3d, omgp3d
  - jpfp3d
- External input file filephs
- External input file filesfc
- External input file filea3d
  - Sequential access format
  - Direct access format

Edit

Namelist integ

jsflx

[integer scalar]
Switch for calculating fluxes (area-averaged fluxes). They are computed as follows:

```
0: no flux output
1: at top and bottom of the atmosphere (TOA and BOA, respectively)
2: at interfaces of all layers except internal interfaces of 3-D inhomogeneous layers
3: at interfaces of all layers
```

The number of interfaces at which fluxes are computed will be

```
0    for jsflx=0,
2    for jsflx=1,
nz - nz3 + 2    for jsflx=2,
nz + 1    for jsflx=3.
```

See also nz and nz3.

jshrt

[integer scalar]
Switch for calculating heating rates. If jshrt=1, then heating rates are computed for all nz layers.

nrdc

[integer scalar]
Number of area-averaged radiances, one of which is specified by altitude and direction.

nxr, nyr

[integer scalar]
Number of pixels along x and y axes, respectively, for pixel-averaged radiances.

therdc, phirdc

[real 1-D array, {dat(irdc), irdc=1, nrdc}]

Radiance zenith and azimuth angles, respectively, in degree. The angles corresponds to the light transport vectors.

jzrdc

[integer 1-D array, {dat(irdc), irdc=1, nrdc}]
Layer interface number that specifies the altitude at which the area-averaged radiances are sampled. Should be lager than or equal to 0 and small than or equal to nz. The interface jzrdc=0 corresponds to the bottom of atmosphere (BOA). If jzrdc > nz, then jzrdc will be reset to nz, by the code. The horizontal location for radiance sampling is defined at the sampling level. Note that this differs from the method that is used for map projection of satellite observation image pixels, which are mapped at the ground level (not at the satellite level).

ncam

[integer scalar]
Number of cameras. One camera image is composed of angular-averaged radiances viewing from a point.

nxc, nyc

[integer scalar]
Numbers of pixels along horizontal and vertical directions, respectively, of camera images.

jzcam

[integer 1-D array, {dat(icam), icam=1, ncam}]
Layer interface number at which each camera is located. Should be between 0 and nz.

xcam, ycam

[real 1-D array, {dat(icam), icam=1, ncam}]
Relative horizontal location (0 to 1) along x and y-axes, respectively, for the camera. The x-coordinate between 0 and xmax corresponds to xcam = 0 to 1. It is also similar for y-coordinate.

rmincam

[real 1-D array, {dat(icam), icam=1, ncam}]
Minimum distance (m) of the view region of the camera. Light sources from the domain closer to the camera is forced to set as it is from the point of the limited distance, rmincam. For simulation of physically-correct images, this limit should be set as 0. However, simulated images will be very noisy when scattering medium is present near the camera. Such a situation frequently appears when Rayleigh scattering is taken into account. For that reason, light from source emission or scattering event near the camera should be adjusted for obtaining smooth, better-looking camera image.

rmidcam

[real 1-D array, {dat(icam), icam=1, ncam}]
Moderate distance (m) for the camera radiance.

rmaxcam

[real 1-D array, {dat(icam), icam=1, ncam}]
Maximum distance (m) of the view region of camera. This limits distance between emission/scattering point and the camera. For simulation of physically-correct images, this limit should be infinity. However, such a simulation is not feasible. See also rmincam.

thecam, phicam

[real 1-D array, {dat(icam), icam=1, ncam}]
Zenith and azimuth angles, respectively, in degree, of direction of the center of camera image. The angles correspond to the camera-to-target vector.

ahcam

[real 1-D array, {dat(icam), icam=1, ncam}]
Maximum view angle (degrees) along camera image's horizontal (scan) direction. The horizontal direction is perpendicular to the vertical direction at the image center.

avcam

[real 1-D array, {dat(icam), icam=1, ncam}]
Maximum view angle (degrees) along camera image's vertical (track) direction. At the image center, a plane along the vertical direction of the image includes the zenith and nadir direction in the spatial domain (the Cartesian coordinate system).

amcam

[real 1-D array, {dat(icam), icam=1, ncam}]
Maximum angle (degrees) of the field of view of the camera, from the center direction. Should be less than 90 degrees.

diacam

[real 1-D array, {dat(icam), icam=1, ncam}]
Diameter (in meter) of the camera lens surface. The surface is assumed as spherical. If diacam=0, then the coordinates of the camera focus point should be (xcam, ycam). Otherwise (diacam >0), the point is at

```
x = xcam - D/2*sin(thecam)*cos(phicam), and
y = ycam - D/2*sin(thecam)*sin(phicam),
```

where D is diameter of the lens surface sphere:

```
D = diacam / sin(amcam).
```

See also thecam, phicam, and amcam.

npot

[integer scalar]
Number of detectors for the local fluxes.

nppot

[integer scalar]
Number of pathlength bins for the local fluxes. See also pminpot and pmaxpot.

nqpot

[integer scalar]
Number of view angle bins for the local fluxes. See also qmaxpot.

jsppot

[integer 1-D array, {dat(ipot), ipot=1, npot}]
Switch for integration of plane flux (jsppot = 0) or spherical flux (jsppot = 1).

jzpot

[integer 1-D array, {dat(ipot), ipot=1, npot}]
Index of layer boundary at which the local flux is sampled. Should be between 0 and nz.

xpot, ypot

[real 1-D array, {dat(ipot), ipot=1, npot}]
Relative horizontal location (0 to 1) along x and y-axes, respectively, for the local flux. The x-coordinate between 0 to xmax corresponds to xpot = 0 to 1. It is also similar for y-coordinate.

rminpot

[real 1-D array, {dat(ipot), ipot=1, npot}]
Minimum distance (m) of the view domain of the local flux. Emission/scattering from regions closer to the detector is forced to set as it is from the point of the limit distance, rminpot. For simulation of physically-correct results, this limit should be 0. However, simulated results will be very noisy if scattering medium is present near the detector. Such a situation frequently appears when Rayleigh scattering is taken into account. For this reason, light sources from emission/scattering events that occurs near the detector should be adjusted for obtaining better-looking results.

rmidpot

[real 1-D array, {dat(ipot), ipot=1, npot}]
Moderate distance (m) for the local flux.

rmaxpot

[real 1-D array, {dat(ipot), ipot=1, npot}]
Maximum distance (m) of the view domain of the local flux. This limits distance between scattering point and the detector. For simulation of physically-correct results, this limit should not be set as infinity. However, such a simulation is highly time-consuming. See also rminpot.

thepot, phipot

[real 1-D array, {dat(ipot), ipot=1, npot}]
Zenith and azimuth angles, respectively, in degree, of direction of the center of the local flux detector. The angles correspond to detector-to-target vector.

pminpot, pmaxpot

[real 1-D array, {dat(ipot), ipot=1, npot}]
Minimum and maximum limits of pathlength for sampled local flux, where the pathlength is integrated flight length of photon between the source emission point and the detector sampling point. The pathlength range is divided into bins, number of which is

nppot.

qmaxpot
[real 1-D array, {dat(ipot), ipot=1, npot}]
Maximum angle (degrees) of the field of view of the local flux detector, from the center direction. The field of view is conical angular region. Note, however, that qmaxpot can be larger than 90 degrees. When qmaxpot=180 degrees, for example, then the field of view is the globe. The angular field is divided into bins, number of which is nqpot.

diapot
[real 1-D array, {dat(ipot), ipot=1, npot}]
Diameter (in meter) of the lens surface of the local flux detector. The detector surface is assumed as spherical. If diapot=0, then the coordinates of the detector focus point should be (xpot, ypot). Otherwise (diapot > 0), the point is at

```
x = xpot - D/2*sin(thepot)*cos(phipot), and
y = ypot - D/2*sin(thepot)*sin(phipot),
```

where D is diameter of the lens surface sphere:

```
D = diapot / sin(qmaxpot).
```

See also thepot, phipot, and qmaxpot.

Edit

Namelist model

nx, ny, nz
[integer scalar]
Numbers of grids along x, y, and z-axis, respectively, in the spatial domain.

iz3l
[integer scalar]
Lowest index of z-coordinate grid (or layer), of horizontally inhomogeneous layers. Layers for the z-index iz=iz3l to iz3l + nz3 - 1 are defined as horizontally inhomogeneous, and for these layers 3-D data of optical properties are given. See also nz3.

nz3
[integer scalar]
Number of horizontally inhomogeneous layers, which is defined for the z-grid index

```
iz=iz3l to iz3l + nz3 - 1.
```

See also iz3l.

xbin, ybin
[real scalar]
Grid spacing along x and y-axis, respectively.

jrfr
[integer scalar]
Switch for atmospheric refraction. If jrfr = 1, then refraction is simulated at each atmospheric layer boundary. Reflection can also occur when the refraction is taken into account. Otherwise (jrfr = 0), the refraction/reflection is neglected.

jsph
[integer scalar]
Switch for correction of atmospheric sphericity effects on radiance and air mass factors. Should be 0 or 1. The correction is implemented only for the line of sight (the path between the last scattering point and the radiance detector). This may influence on results of radiance of direction near the horizon (within 3 degrees) under some special conditions (e.g., very optically thin atmosphere).

rplanet
[real scalar]
Radius (in meter) of the planet, for the sphericity correction on the radiances and air mass factors. Used only when jsph=1. The planet surface should correspond to z = 0. For the earth, the radius is 6.74e+6 m approximately.

jtmplev
[integer scalar]
Defines a method to specify temperature data, as follows:

```
0: temperature data are given for layers (grids), and the temperature is assumed as constant in each layer.
```

```
1: temperature data are given for layer interfaces, and the temperature is
```
assumed to vary linearly along z-coordinate in each layer. Required number of z-axis levels for temperature data is nz for jtmplev=0 and nz+1 for jtmplev=1. Temperature data are used only when the source includes thermal emission (jdefsrc = 2 or 3).

zgrd
[real 1-D array, {dat(iz), iz=0, nz}]
Z-coordinate values (in meter) of layer boundaries. The top and bottom of a layer with index iz corresponds to z=zgrd(iz) and zgrd(iz+1), respectively. The BOA (surface) corresponds to zgrd(0), and TOA zgrd(nz).

drfr
[real 1-D array, {dat(iz), iz=1, nz}]
Anomalies in atmospheric refractive index from unity, for each layer. Used only when jrfr=1. Give null if the index is unity. Actual values of refractive indexes should be 1+drfr.

dens
[real 1-D array, {dat(iz), iz=1, nz}]
Gas density (in relative unit) for air mass factor calculations, used only when calculating area-averaged radiance. This is used as a weighting function for average air mass factors of radiance.

tmpa1d
[real 1-D array, {dat(iz), iz=1, nz}] when jtmplev=0
[real 1-D array, {dat(iz), iz=0, nz}] when jtmplev=1
Horizontally-averaged atmospheric temperature [K] defined at layers if jtmplev=0, or at layer interfaces if jtmplev=1. Used for thermal emission when jdefsrc = 2 or 3. When jtmplev=1, an interface of iz=0 corresponds to BOA, and that of iz=nz corresponds to TOA. Note that tmpa1d is atmospheric temperature even when iz=0 (BOA). The surface temperature should be given in namelist mdlsfc. For horizontally inhomogeneous layer, data of perturbation in temperature will be specified in namelist mdla3d.

ext1st, omg1st
[real 1-D array, {dat(iz), iz=1, nz}]
Extinction coefficients (/m) and single scattering albedo, respectively, of the first kind optical medium, which can be gas, aerosol, hydrometeor, or their mixture. Note that all optical properties are assumed as constant in every cells.

jpf1st
[integer 1-D array, {dat(iz), iz=1, nz}]
indexes of phase functions of the first kind scattering medium. Should be between 0 and npf. If jpf1st = 0, then predefined Rayleigh phase function is used.

ext2nd, omg2nd, jpf2nd
[real 1-D array, {dat(iz), iz=1, nz}]
As ext1st, omg1st, and jpf1st, respectively, but for the second kind optical medium.

ext3rd, omg3rd, jpf3rd
[real 1-D array, {dat(iz), iz=1, nz}]
As ext1st, omg1st, and jpf1st, respectively, but for the third kind optical medium.

nkd
[integer scalar]
Number of terms for the k-distribution used as the gaseous absorption model. Give nkd=1 for monochromatic computation.

wkd
[real 1-D array, {dat(ikd), ikd=1, nkd}]
Weight coefficients of the correlated k-distribution. Give wkd(1)=1 for monochromatic computation.

absg1d
[real 2-D array, {(dat(ikd, iz), ikd=1, nkd), iz=1, nz}]
Horizontally-averaged gaseous absorption coefficients (/m) of the correlated k-distribution. Perturbations of the absorption coefficient in the horizontally inhomogeneous layers can be given in namelist mdla3d.

Edit

Namelist mdlphs

filephs
[character string scalar]
Filename for external file input of phase function data. Leave it as default ' ' if no external input file is needed. The name should have relative path from a directory where the configuration file exists. The external file input will override on input from the following

arguments of namelist mdlphs.

npf
[integer scalar]
Number of phase functions

nang
[integer 1-D array, {dat(ipf), ipf=1, npf}]
Numbers of angle grids for the phase function. The followings are exceptions:

- If nang=1, then Heyney-Greenstein phase function with asymmetry parameter value of phs(iang, ipf) for iang=1.
- If nang=0, then Rayleigh scattering phase function is used.

ang
[integer 2-D array, {(dat(iang, ipf), iang=1, nang), ipf=1, npf}]
Angles in degree for the phase function.

phs
[integer 2-D array, {(dat(iang, ipf), iang=1, nang), ipf=1, npf}]
Phase functions. Relative values are acceptable, because the phase functions are automatically normalized in the code. See also nang.

Edit

Namelist mdlsfc

filesfc
[character string scalar]
Filename for external file input of surface model data. Leave it as default ' ' if no external input file is needed. The name should have relative path from a directory where the configuration file exists. The external file input will override on input from the following arguments of namelist mdlsfc. The file can be the same as filephs.

nxb, nyb
[integer, scalar]
Numbers of (X, Y) pixels for the surface properties. This is now independent of the atmosphere. For example, if nxb = 1 and nyb = 1, then surface properties are constant in the full domain.

tmps2d
[real 2-D array, {(dat(ixb, iyb), ixb=1, nxb), iyb=1, nyb}]
The surface temperature (in K). Used for calculation of thermal emission when jdefsrc = 2 or 3.

jsfc2d
[integer 2-D array, {(dat(ixb, iyb), ixb=1, nxb), iyb=1, nyb}]
Index for surface BRDF model of each pixel at (ixb, iyb):

```
0: black
1: Lambertian
2: diffuse-specular mixture (DSM) model
3: Rahman-Pinty-Verstraete (RPV) model
4: Li-Sparse-Ross-Thick (LSRT) model
```

This is neglected if jsfc2d is given in the file, filesfc.

psfc2d
[real 2-D array, {((dat(ixb, iyb, ip), ixb=1, nxb), iyb=1, nyb), ip=1, 5}]
Surface parameters of each pixel at (ixb, iyb). These are neglected if psfc2d is given in the file, filesfc. Definitions of psfc2d for respective ip differ by the BRDF model (jsfc2d), as follows:

- jsfc2d = 0 (black): No specification is required for psfc2d, which is dummy parameter.
- jsfc2d = 1 (Lambertian):
  - psfc2d(ip=1): albedo
- jsfc2d = 2 (DSM): The surface reflection is modeled by a microscopical mixture of diffusive and specular reflectors.
  - psfc2d(ip=1): diffuse albedo (a_d)
  - psfc2d(ip=2): fraction of diffuse reflectors (f_d); Fraction of specular reflection is 1 - psfc2d(ip=2)
  - psfc2d(ip=3): real part of refractive index of underlying material (e.g., water); Should be given as positive.
  - psfc2d(ip=4): imaginary part of refractive index of underlying material (e.g., water); Should be given as positive.

psfc2d(ip=5): variance of micro-scopic surface slope (tangent of the inclination angle). Give zero for flat surface. This is used for specular reflection. According to Cox and Munk's (1954) parameterization for ocean surface,

```
psfc2d(ip=5) = 0.00512*V + 0.003.
```

where V is wind velocity (m/s) at 10 m above the ocean surface.
The DSM model albedo follows

```
a = f_d * a_d + (1 - f_d) * a_s,
```

where a_s is albedo for specular reflection. BRDF of the DSM model follows the similar equation.

- jsfc2d = 3 (RPV): Rahman-Pinty-Verstraete (RPV) BRDF model
  - psfc2d(ip=1): the first parameter, alpha
  - psfc2d(ip=2): the exponent parameter, k
  - psfc2d(ip=3): the asymmetry parameter, THETA
  - psfc2d(ip=4): delta

The RPV BRDF, R, is given as

```
pi * R = alpha * [u0 * u1 / (u0 + 1) / (u1 + 1)]^(k - 1) * HG(THETA) * F(G)
```

where pi=3.141592653, u0 and u1 are respectively cosines of incidence and reflection vectors, and HG is the Heyney-Greenstein function. The function F is as follows:

```
F(G) = 1 + (1 + G)/(delta + G),
```

where G is a geometrical parameter.

- jsfc2d = 4 (LSRT): Li-Sparse-Ross-Thick (LSRT) BRDF model
  - psfc2d(ip=1): "k_L", weight of Lambertian reflection
  - psfc2d(ip=2): "k_g", weight of geometrical optics reflection
  - psfc2d(ip=3): "k_v", weight of volume scattering

The LSRT BRDF, R, is given by

```
pi * R = k_L * pi + k_g * K_geo + k_v * K_vol
```

where K_geo and K_vol are geometrical and volume scattering kernel functions.

Edit

Namelist mdla3d

ifmt3d

[integer scalar]
Switch for format of the external input file, filea3d:

```
0: Sequential-access text format,
1: Direct-access binary format.
```

npar

[integer scalar]
Number of scattering media in horizontally inhomogeneous layers. A medium can be a mixture of gases, aerosols and hydrometeors or a size-bin of their size distributions.

filea3d

[character string scalar]
Filename of an external file of 3-D atmospheric model data. Leave it as default ' ' if no external input file is needed. The name should have relative file name path from the directory where the configuration file exists. The external file input will override on input from the following arguments of namelist mdla3d (except ifmt3d and npar). The file can be the same as filephs and/or filesfc when sequential access format (ifmt3d=0).

tmpa3d

[real 3-D array, {((dat(ix, iy, iz3), ix=1, nx), iy=1, ny), iz3=1, nz3}] when jtmplev=0
[real 3-D array, {((dat(ix, iy, iz3), ix=1, nx), iy=1, ny), iz3=0, nz3}] when jtmplev=1
3-D distributions of perturbations of atmospheric temperature [K] at layers (jtmplev=0) or layer interfaces (jtmplev=1) of horizontally inhomogeneous layers. Used for calculation of thermal emission when jdefsrc = 2 or 3. When jtmplev=0, an interface of index iz3=0 corresponds to the bottom of the lowest inhomogeneous layer, and iz3=nz3 corresponds to the top of the highest inhomogeneous layer. Actual temperatures are represented as follows,

```
temperature(ix, iy, iz) = tmpald(iz) + tmpa3d(ix, iy, iz3), where iz = iz3l + iz3 - 1.
```

See also tmpa1d and iz3l.

absg3d

[real 4-D array, {((((dat(ix, iy, iz3, ikd), ix=1, nx), iy=1, ny), iz3=1, nz3), ikd=1, nkd}]
3-D distributions of perturbations of gaseous absorption coefficient of the correlated k-distribution. Actual absorption coefficients are represented as follows,

```
coefficient(ix, iy, iz, ikd) = absgld(iz, ikd) + absg3d(ix, iy, iz3, ikd),
```

where iz = iz3l + iz3 - 1. Note that the total coefficient (absg1d + absg3d) may not be less than zero.

extp3d, omgp3d

[real 4-D array, {((((dat(ix, iy, iz3, ipar), ix=1, nx), iy=1, ny), iz3=1, nz3), ipar=1, npar}]
Extinction coefficients and single scattering albedos, respectively, of media in horizontally inhomogeneous layers with iz3=1 to nz3. A medium can be a mixture of gases, aerosols and hydrometeors or a size-bin of their size distributions.

jpfp3d

[integer 4-D array, {((((dat(ix, iy, iz3, ipar), ix=1, nx), iy=1, ny), iz3=1, nz3), ipar=1, npar}]
Indexes of phase functions of media in horizontally inhomogeneous layers. Should be between 0 and npf. If jpfp3d=0, then predefined Rayleigh phase function is used.

Edit

External input file filephs

The file for phase function data should be sequential access file. The data block to be read is recognized after a line containing the following "signature":

```
%mdlphs
```

After the signature is found, the data are read by the following Fortran codes;

```
read (iu, *) npf   ! number of phase functions
 do ipf = 1, npf
   read (iu, *)
   read (iu, *) nang(ipf)     ! number of angles
   read (iu, *)
   do iang = 1, nang(ipf)
     read (iu, *) ang(iang, ipf), phs(iang, ipf)    ! angle in degree, and phase function
    end do
 end do
```

Example data files can be found in the "examples" directory. See also namelist mdlphsfor details of each variable.

Edit

External input file filesfc

The file for surface property data should be sequential access file. The data block to be read is recognized after a line that contains the following line:

```
%mdlsfc
```

After the signature is found, the data are read by the following Fortran codes;

```
          read (iu, *)
          read (iu, *) ((tmps2d(ixb, iyb), ixb = 1, nxb), iyb = 1, nyb) ! temperature
          read (iu, *)
          read (iu, *) ((jsfc2d(ixb, iyb), ixb = 1, nxb), iyb = 1, nyb) ! surface BRDF model index
          read (iu, *, end=1)
          do ip = 1, 5
             read (iu, *, err=1, end=1) ((psfc2d(ixb, iyb, ip), ixb = 1, nxb), iyb = 1, nyb)
             !// surface parameters
          end do
 1     continue
```

Example data files can be found in the "examples" directory. See also namelist mdlsfcfor details of each variable.

Edit

External input file filea3d

The file for 3-D atmospheric property data should be sequential access text format if ifmt3d=0 or direct access binary format if ifmt3d=1.

Sequential access format

When sequential access text format (ifmt3d=0), the data block to be read is recognized after a line that contains the following line:

```
%mdlphs
```

After the signature is found, the data are read by the following Fortran codes:

```
      read (iu, *)
      if (jtmplev .eq. 0) then
         iz3min = 1
      else
         iz3min = 0
      end if
      do iz3 = iz3min, nz3
            read (iu, *) ((tmpa3d(ix, iy, iz3), ix = 1, nx), iy = 1, ny) ! temperature
      end do
      do ikd = 1, nkd
         read (iu, *)
         do iz3 = 1, nz3
               read (iu, *) ((absg3d(ix, iy, iz3, ikd), ix = 1, nx), iy = 1, ny)
               !// gas absorption coefficient
         end do
      end do
      do ipar = 1, npar
         read (iu, *)
         do iz3 = 1, nz3
               read (iu, *) ((extp3d(ix, iy, iz3, ipar), ix = 1, nx), iy = 1, ny)
               !// extinction coefficient
         end do
         read (iu, *)
         do iz3 = 1, nz3
               read (iu, *) ((omgp3d(ix, iy, iz3, ipar), ix = 1, nx), iy = 1, ny)
               !// single scattering albedo
         end do
         read (iu, *)
         do iz3 = 1, nz3
            read (iu, *) ((rpfp3d(ix, iy), ix = 1, nx), iy = 1, ny)
            !// index of phase function
            do iy = 1, ny
               do ix = 1, nx
                  jpfp3d(ix, iy, iz3, ipar) = int(rpfp3d(ix, iy) + 0.00001)
                  !// real-to-integer conversion
               end do
            end do
         end do
      end do
```

Note that in the last part of the above code, the real values for rpfp3d are rounded to integer values for jpfp3d. See also namelist mdla3d for details of each variable. Example data files can be found in the "examples" directory.

Direct access format

When direct access binary format (ifmt3d=1), record length is set as nrec=nbyte*nx*ny. The data are read as the following Fortran codes:

```
      irec = 1
      if (jtmplev .eq. 0) then
         iz3min = 1
      else
         iz3min = 0
      end if
      do iz3 = iz3min, nz3
         read (iu, rec=irec) ((tmpa3d(ix, iy, iz3), ix = 1, nx), iy = 1, ny) ! temperature
         irec = irec + 1
      end do
      do ikd = 1, nkd
         do iz3 = 1, nz3
            read (iu, rec=irec) ((absg3d(ix, iy, iz3, ikd), ix = 1, nx), iy = 1, ny)
            !// gas absorption coefficient
            irec = irec + 1
```

```
      end do
   end do
   do ipar = 1, npar
      do iz3 = 1, nz3
         read (iu, rec=irec) ((extp3d(ix, iy, iz3, ipar), ix = 1, nx), iy = 1, ny)
         !/// extinction coefficient
         irec = irec + 1
      end do
      do iz3 = 1, nz3
         read (iu, rec=irec) ((omgp3d(ix, iy, iz3, ipar), ix = 1, nx), iy = 1, ny)
         !/// single scattering albedo
         irec = irec + 1
      end do
      do iz3 = 1, nz3
         read (iu, rec=irec) ((rpfp3d(ix, iy), ix = 1, nx), iy = 1, ny)
         !/// index of phase function
         do iy = 1, ny
            do ix = 1, nx
               jpfp3d(ix, iy, iz3, ipar) = int(rpfp3d(ix, iy) + 0.00001)
               !/// real-to-integer conversion
            end do
         end do
         irec = irec + 1
      end do
   end do
```

Edit

Next

- 5. Radiative Transfer (RT) Simulation
  - Examples
  - Usage of "mcarats" executables
  - Possible problems
  - Tips using tools
  - Quick visualization using GrADS
- 6. Output data from the "mcarats" codes
  - Format
  - Header section
  - Data section
  - Reading the output

Edit

5. Radiative Transfer (RT) Simulation

Examples

The user can easily learn about how to use the RT simulation tools from a shell script, ./examples/Job.csh, and the example I/O files in ./examples.

Usage of "mcarats" executables

The following executables will be installed in ./bin directory after the installation:

```
mcarats3d: for a 3-D model atmosphere
mcarats2d: for a 2-D model atmosphere
mcarats1d: for a 1-D model atmosphere
```

The usage of the RT simulation tools is, for example, as follows:

```
% ./bin/mcarats3d ptot solver recomp configfile newoutfile (oldoutfile)
ptot    : Total number of incident photons. The format is integer or real (e.g., 1000000 or 1.0e+6)
solver  : Radiative transfer solver
  =0: F3D, Fully-3-Dimensional radiative transfer
  =1: P3D, Partially-3-Dimensional radiative transfer
  =2: ICA, Independent Column (or pixel) Approximation
recomp    : Recomputation switch
  =0: New experiment without previous results: Q = Qnew
  =1: Recomputation/average: Q = f*Qnew + (1-f)*Qold, where f=ptot_new/(ptot_new + ptot_old)
  =2: Integration/adding: Q = Qnew + Qold
configfile: Configuration file containing the namelist input
newoutfile: New output file
oldoutfile: Old file containing precomputed results (optional)
```

For example, when the user operates the following,

```
% ./bin/mcarats3d 1e5 0 0 ./examples/case3/conf_rs0067 out
```

then the output data file, ./out, will be generated. Next, this simulation results can be improved by adding photons:

```
% ./bin/mcarats3d 1e5 0 1 ./examples/case3/conf_rs0067 out2 out
```

The new output file, ./out2, contains results for total 200,000 incident photons. This recomputation process can be done recursively, and this function may be useful in some cases.

Possible problems

1) Invalid input format
The most frequent error is caused by invalid input format of the configuration file that contains namelist parameters. Format for namelist input may slightly differ by Fortran compiler. Please read carefully the compiler manual for valid namelist input format. In addition, if the code reported reading error of namelist input or external input files, then check the configuration file or the external input files is written in valid format as described here.
2) Invalid input values
The other common error is from invalid input values. In this case, diagnostics might be reported from the codes (in some cases). Change the values for the input variables in valid ranges. Especially, input values for a variable should be integer if the variable is declared as integer. In addition, some variables should be in specific ranges. Check also whether the arguments of each array variable are not out of bounds. See the descriptions of the input variables.
3) Output results are noisy
This may be due to small number of trajectories simulated. Increase the number ptot, in this case.
5) Bug?
In other cases, there is no advise from the developers. Please feel free to ask any question to developers, who hope to reply to all

questions as long as there is enough time. If you find any bug, please report to the developers.

Tips using tools

To simulate for several similar cases with small differences, there is a useful tool,

```
./shl/replconfig.awk
```

that is an awk script. This tool can replace one parameter value in the input configuration file. A limitation is that the variable to be replaced should be scalar (i.e., not array variable). For example, we can replace the value for variable fmax in

```
./examples/case3/conf_rs0067
```

by

```
% awk -f ./shl/replconfig.awk fmax 0.6 < ./examples/case3/conf_rs0067 > ./conf
```

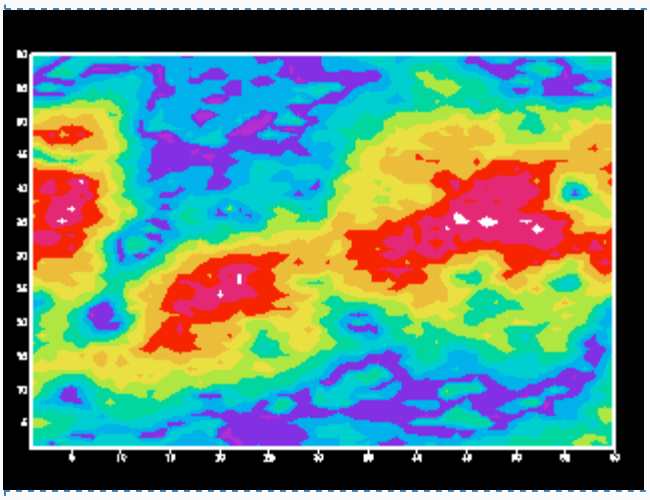
Then the new ./conf file contains a line as "fmax = 0.6". This is useful to perform many simulations, as follows:

```
% awk -f ./shl/replconfig.awk fmax 0.0 < ./examples/case3/conf_rs0067 > ./conf
% ./bin/mcarats3d 1e5 0 0 ./conf ./out-fmax0.0
% awk -f ./shl/replconfig.awk fmax 0.2 < ./examples/case3/conf_rs0067 > ./conf
% ./bin/mcarats3 1e5 0 0 ./conf ./out-fmax0.2
% awk -f ./shl/replconfig.awk fmax 0.4 < ./examples/case3/conf_rs0067 > ./conf
% ./bin/mcarats3d 1e5 0 0 ./conf ./out-fmax0.4
% awk -f ./shl/replconfig.awk fmax 0.6 < ./examples/case3/conf_rs0067 > ./conf
% ./bin/mcarats3d 1e5 0 0 ./conf ./out-fmax0.6
% awk -f ./shl/replconfig.awk fmax 0.8 < ./examples/case3/conf_rs0067 > ./conf
% ./bin/mcarats3d 1e5 0 0./conf ./out-fmax0.8
% rm -f ./conf
```

Quick visualization using GrADS

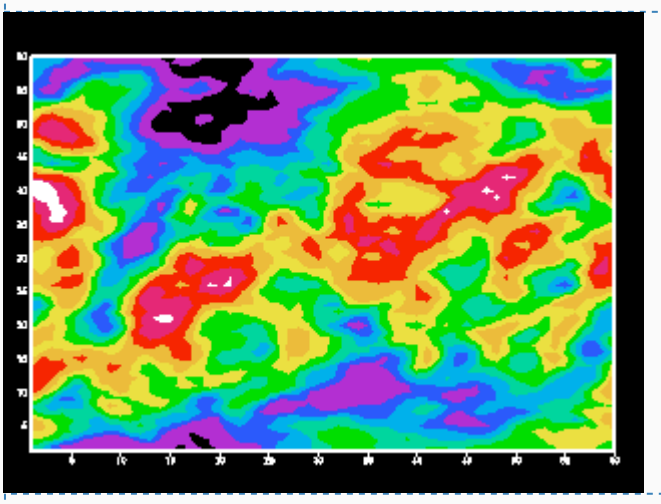
To quickly visualize output data from the RT codes ("mcarats*"), a tool, pgrads, is provided. This converts the output data file to GrADSformat files. If GrADS is installed in your system, you can try the followings:

```
% ./bin/mcarats3d 1e5 0 0 ./examples/case3/conf_rs0067 out
% ./bin/pgrads 1 3 0 ./out
% grads -l
...
ga-> open out.gd.ctl
Scanning description file:  out.gd.ctl
Data file out.gd is open as file 1
LON set to 1 64
LAT set to 1 64
LEV set to 1 1
Time values set: 2000:1:0:0 2000:1:0:0
ga-> set gxout shaded
ga-> set mproj off
ga-> set clevs 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Number of clevs = 11
ga-> d rad
Contouring at clevs =  0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
```



```
ga-> set t 2
Time values set: 2001:1:0:0 2001:1:0:0
ga-> set clevs 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
```

```
Number of clevs = 11
ga-> d rad
Contouring at clevs =  0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
```



```
ga-> quit
No hardcopy metafile open
GX package terminated
```

When "d rad" command is entered, the window displays images. Details of pgrads command are written here.

Edit

6. Output data from the "mcarats" codes

Format

The general format of "mcarats" output file is as follows:

```
header
data
header
data
header
data
...
```

For example, see ./examples/case3/out_rs0067-f3d. A set of one header section and one data section corresponds to one definition of radiation source in the input configuration file (see ./examples/case3/conf_rs0067). Thus the number of sets of the header and data sections is equal to "nsrc" in the configuration file.

Header section

At the beginning of the output file, header information are recorded. The following is an example:

```
%mcarats-0.9
    1 : isrc
    0 : isolver
1.00000E+09 : ptot
   64    64     0    0 : nx, ny, nzf, nzh
   64    64     1 : nxr, nyr, nrdc
  600   450     0 : nxc, nyc, ncam
  513   257     0 : nxp, nyp, npot
```

Notations are as follows:

- solver: radiative transfer solver; 0 for Fully 3-D (F3D) scheme, 1 for Partially-3-D (P3D) scheme, and 2 for ICA
- ptot: total number of source photons.
- nx, ny: numbers of x and y-grids, respectively, for fluxes and heating rates
- nzf: number of z-grids for fluxes
- nzh: number of z-grids for heating rates
- nxr, nyr: numbers of x and y-grids, respectively, for radiances
- nrdc: number of definitions of radiances
- nxc, nyc: numbers of x and y-grids, respectively, for camera image radiances
- ncam: number of cameras

nppot, nqpot: numbers of pathlength bins and FOV angle bins, respectively, for the local fluxes

-
- npot: number of specifications of local flux

Fortran 77 source codes to read/write the header are provided in the package:

```
./src/libmcarats/getmchdr.f and ./src/libmcarats/putmchdr.f.
```

Data section

The data section contains five subsections for fluxes, heating rates, radiances, camera radiances, and local fluxes. Each subsections are written by codes such like the followings:

- Flux subsection:

```
if (nzf .gt. 0) then
            do iflx = 1, 3
               write (iuo, '(a, i6)', err=101) '# iflx= ', iflx
               do izf = 1, nzf
                  write (iuo, *) ((pflx(ix, iy, izf, iflx), ix = 1, nx), iy = 1, ny)
               end do
            end do
         end if
```

- Heating rate subsection:

```
if (nzh .gt. 0) then
               write (iuo, '(a)', err=101) '# heating rate'
               do izh = 1, nzh
                  write (iuo, *) ((phrt(ix, iy, izh), ix = 1, nx), iy = 1, ny)
               end do
         end if
```

- Radiance & air mass factor) subsection:

```
do imom = 1, 2
    do irdc = 1, nrdc
        write (iuo, '(a, 2i6)', err=101) '# irdc, imom: ', irdc, imom
        write (iuo, *) ((prdc(ixr, iyr, imom, irdc), ixr = 1, nxr), iyr = 1, nyr)
    end do
 end do
```

Note that prdc is radiance (imom=1) or AMF (imom=2).

- Camera radiance subsection:

```
do icam = 1, ncam
               write (iuo, '(a, i6)', err=101) '# icam= ', icam
               write (iuo, *) ((pcam(ixc, iyc, icam), ixc = 1, nxc), iyc = 1, nyc)
            end do
```

- Local flux subsection:

```
do ipot = 1, npot
               write (iuo, '(a, 2i6)', err=101) '# iopot, ipot= ', 1, ipot
               write (iuo, *) ((ppot(ippot, iqpot, 1, ipot), ippot = 1, nppot), iqpot = 1, nqpot)
               !/// contributions from all events
               write (iuo, '(a, 2i6)', err=101) '# iopot, ipot= ', 2, ipot
               write (iuo, *) ((ppot(ippot, iqpot, 2, ipot), ippot = 1, nppot), iqpot = 1, nqpot)
               !/// contributions from source emission plus single scattering
            end do
```

See, for example,

```
./examples/case3/out*.
```

Reading the output

The output files can be read in the same way in ./src/process/pconvert.f, for example.

Next

- 7. Post-processing
  - Introduction
  - pcouple
  - pconvert
  - pgrads
  - psample
  - pstat

Edit

7. Post-processing

Introduction

The output file of mcarats codes is in a unique format. Several tools are provided to process output data files from mcarats codes. The user can easily make a new tool using subroutines in ./src/libmcarats/. If you made a useful tool, you can contribute to the project by providing it.

- pcouple to do a simple calculation with a couple of mcarats output data files
- pconvert to convert a mcarats output data file with a simple equation
- pgrads to visualize a mcarats output data set using GrADS
- psample to (re)sample data from mcarats output file
- pstat to calculate spatial statistics of mcarats output data

pcouple

This is used to do a simple calculation with a couple of mcarats output data files.

```
usage: pcouple method thresh infile1 infile2 > outfile
```

- method:

```
    0 (sum)               : in1 + in2
    1 (difference)        : in1 - in2
    2 (product)           : in1 * in2
    3 (ratio)               : in1 / in2
    4 (relative diff. 1) : (in1 - in2) / in2
    5 (relative diff. 2) : 2*(in1 - in2) / (in1 + in2)
    6 (square diff.)       : in1*in1 - in2*in2
    7 (root square diff.): sqrt(in1*in1 - in2*in2)
    8 (diff. square)       : (in1 - in2)*(in1 - in2)
    9 (root diff.)          : sqrt(in1) - sqrt(in2)
```

- thresh: threshold of the denominator when method=3-5. If the denominator is less than the threshold, then the output value is reset at a very small value (-1.0E+30).
- infile1 : Input data file 1.
- infile2 : Input data file 2.
- outfile : Output data file.

The output file is in the same format as the input.

pconvert

This is used to convert a mcarats output data file with a simple equation.

```
usage: pconvert nx ny method A B < infile > outfile
```

- nx : # of x-pixels for output.
- ny : # of y-pixels for output.

If nx*ny=0, then these are reset as defined in the input file. Otherwise the averaging is operated.

- method=0 : v = u (does nothing)

```
        1 : v = A + B * u       (linear)
        2 : v = A * u^B         (power-law)
        3 : v = exp(A + B*u)    (exponential)
        4 : v = ln(A + B*u)     (logarithmic)
        5 : v = A + (1 + B*G)*u (random noises are superimposed)
            where G is a normalized Gaussian noise (average: 0; variance:1).
```

The output file is in the same format as the input, but if nx or ny is smaller than the input ("nx or ny; nxr or nyr; nxc or nyc; nppot or nqpot" defined in infile itself), then the amount of output data is reduced.

pgrads

To quickly visualize output data from mcarats codes, a tool, pgrads, is provided. This converts mcarats output data file to GrADS format.

```
usage: pgrads mach q zset infile (outfile)  (< zlevels)
```

- mach : machine parameter as follows:

```
   =0   : little endian, record length for a real*4 datum = 4 (x86+g77)
   =1   : little endian, record len = 1 (Alpha/x86+ifort)
   =2   : big    endian, record len = 4 (PPC+g77)
   =3   : big    endian, record len = 1
```

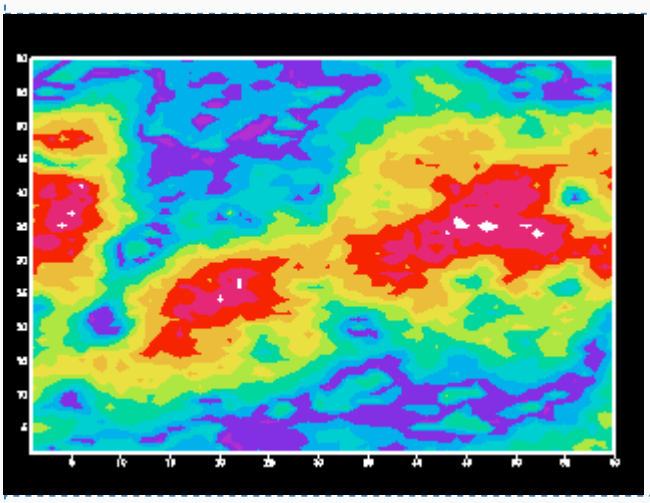
For example

```
   little endian: x86, Alpha,...
   big    endian: PowerPC, Sun,...
   record length=4: g77
   record length=1: ifort
```

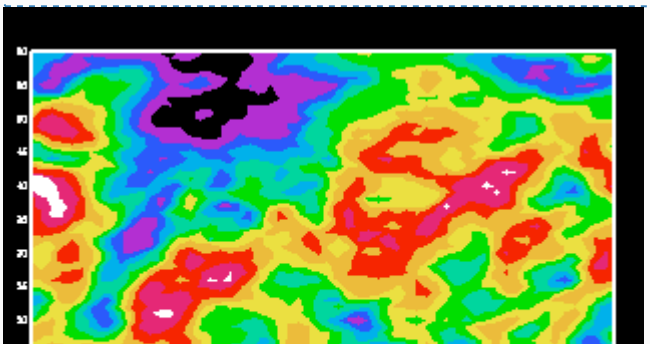
- q : determines which data sets are to be output (0:flux&HR, 1:flux, 2:HR, 3:radiance, 4:camera, 5:local flux)
- zset : method to set z-levels (0: default; 1: read from stdin)
- infile : input file (MCARaTS output)
- outfile : output file (GrADS format)
- zlevels : z-levels when zset=1

If GrADS is installed in your system, you can try the followings (in the "examples/case3" directory):

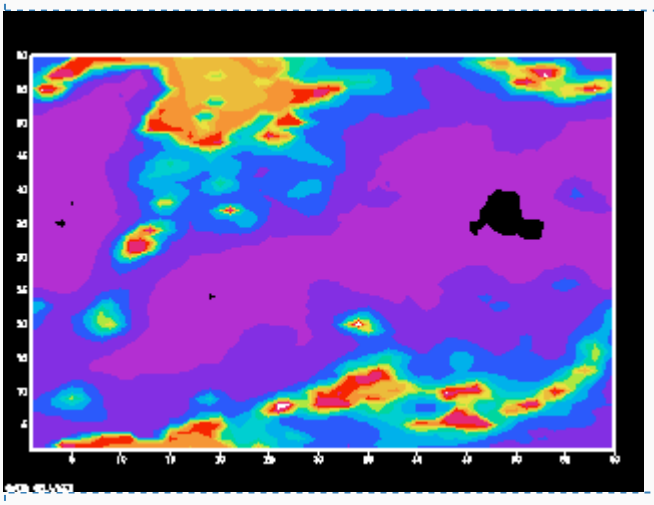
```
% ../../bin/pgrads 1 3 0 out_rs0067-f3d
% grads -l
...
ga-> open out_rs0067-f3d.gd.ctl
...
ga-> set mproj off
ga-> set gxout shaded
ga-> d rad
```



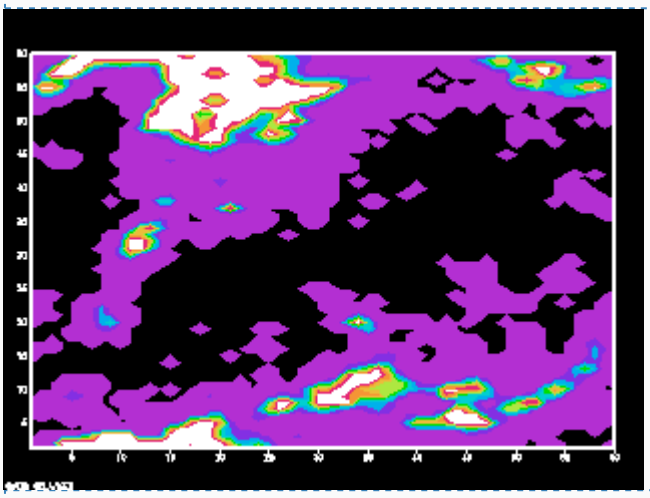
```
ga-> printim rs0067-f3d-t1.png x400 y300
ga-> set t 2
Time values set: 2001:1:0:0 2001:1:0:0
ga-> d rad
```



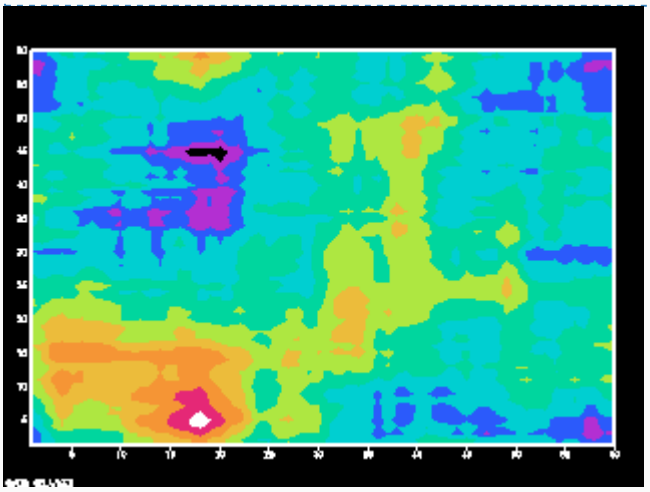
```
ga-> printim rs0067-f3d-t2.png x400 y300
ga-> quit
% ../../bin/pgrads 1 0 0 out_hr0213-f3d
% grads -l
...
ga-> open out_rs0213-f3d.gd.ctl
...
ga-> set mproj off
ga-> set gxout shaded
ga-> d fd
```



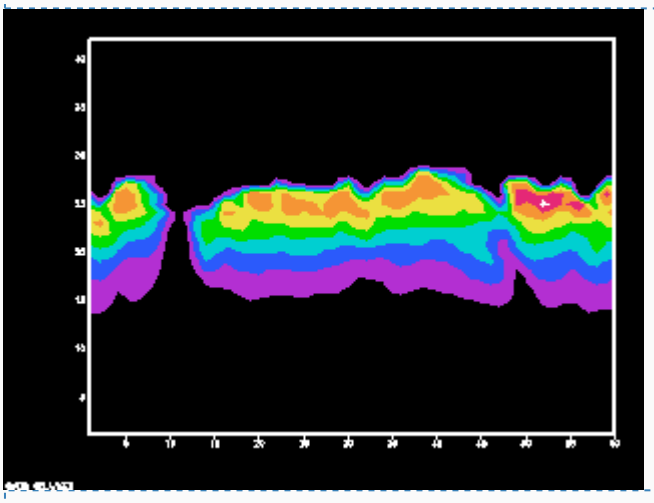
```
ga-> printim hr0213-f3d-t1-fd.png x400 y300
ga-> d f0
```



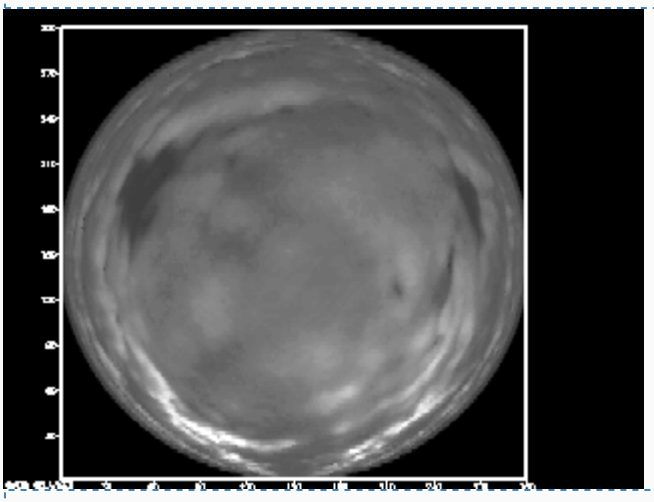
```
ga-> printim hr0213-f3d-t1-f0.png x400 y300
ga-> set z 2
LEV set to 2 2
ga-> d fu
```



```
ga-> printim hr0213-f3d-t1-fu.png x400 y300
ga-> set z 1 42
LEV set to 1 42
ga-> set y 30
LAT set to 30 30
ga-> c
ga-> d hr
```



```
ga-> printim hr0213-f3d-t1-hr.png x400 y300
ga-> quit
% ../../bin/pgrads 1 4 0 out_ci0067-f3d
% grads -l
...
ga-> open out_ci0055-f3d.gd.ctl
ga-> set mproj off
ga-> set parea 1 9 0.2 8.2
ga-> set gxout shaded
ga-> set display grey
ga-> set grid off
grid is off
ga-> set t 1
Time values set: 2001:1:0:0 2001:1:0:0
ga-> set clevs 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 1.4
Number of clevs = 15
ga-> d cam
```



```
ga-> printim ci0055-f3d-t1.png x400 y400
ga-> quit
```

psample

This is used to resample data from a mcarats output data file. This is useful to extract partial data from the file.

```
usage: psample ixi ixs nx iyi iys ny < infile > outfile
```

- ixi, iyi : initial indexes for X and Y

ixs, iys : step of X and Y indexes

-
- nx, ny : # of X and Y pixels for output

An example is here: at examples/case3/

```
% ../../bin/psample 2 2 30 2 2 30 < out_rs0067-f3d > sample_rs0067-f3d
```

In this case, original data for 60x60 pixels are available, and output result from the above is for 30x30 pixels.

pstat

This is used to calculate basic statistics of a text data sequence.

```
usage: pstat N func ithresh (threshold1 threshold2) < infile > outfile
```

- N : # of data
- func : 0, raw data; 1, ln(dat); 2, exp(dat); The conversion is performed before the statistics computation.
- ithresh

```
=0 : no threshold,
=1 : dat >= thresh1,
=2 : dat <= thresh1,
=3 : thresh1 <= dat <= thresh2,
=4 : dat <= thresh1 or dat >= thresh2
```

Output data file reports fraction of data filtered by the threshold(s), minimum, maximum, average, standard deviation, root-mean-square (rms), and skewness. For example,

```
% bin/pstat 3600 0 0 < examples/case3/out_rs0067-f3d
#frac,  min,  max,  average,  stdev,  rms,  skewness
1.0000   3.3890E-01   6.4152E-01   4.7632E-01   7.4342E-02   4.8209E-01   2.0645E-01
1.0000   2.0148E+00   2.0329E+00   2.0212E+00   2.4843E-03   2.0212E+00   8.3968E-01
1.0000   1.3267E-01   6.6358E-01   3.6322E-01   1.1587E-01   3.8126E-01  -1.3765E-01
1.0000   3.0360E+00   3.1539E+00   3.0686E+00   1.9938E-02   3.0687E+00   1.1165E+00
```

The output are printed to stdout. Statistics for four data sequences, each of which contains 3600 data, are reported in this example.

Edit

Return